**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра автоматики та управління в технічних системах**

«До захисту допущено»
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Програмне забезпечення інформаційно-комунікаційних систем»**
**спеціальності 121 «Інженерія програмного забезпечення»**
**на тему: «Застосунок моделювання цифрових фільтрів на основі .Net»**

Виконав:
студент IV курсу, групи ІТ-61
Мішин Олександр Володимирович _____

Керівник:
Професор каф. АУТС, д.е.н.,
Шемаєв Володимир Миколайович _____

Рецензент:
С.н.с., ІТГІП НАН України, д.т.н
Биченок Микола Миколайович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра автоматики та управління в технічних системах**

Рівень вищої освіти — перший (бакалаврський)

Спеціальність — 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

**ЗАВДАННЯ**

**на дипломний проєкт студенту**

**Мішин Олександр Володимирович**

1. Тема проєкту «Застосунок моделювання цифрових фільтрів на основі .Net», керівник проєкту Шемаєв Володимир Миколайович, професор кафедри АУТС, затверджені наказом по університету від «7» травня 2020 р. №1081-с

2. Термін подання студентом проєкту 9 червня 2020 р.

3. Вихідні дані до проєкту: мова програмування C#, середовище розробки Microsoft Visual Studio, технологія WPF

4. Зміст пояснювальної записки: аналіз предметної області, розробка програмного забезпечення, аналіз якості та тестування, керівництво користувача

5. Перелік графічного матеріалу: діаграма класів, діаграма прецедентів, блок-схема алгоритму ШПФ, діаграма активності

6. Дата видачі завдання 3 березня 2020 року

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1. | Дослідження теоретичних відомостей | 10.04.2020 | Виконано |
| 2. | Вибір моделі | 13.04.2020 | Виконано |
| 3. | Проектування системи | 20.04.2020 | Виконано |
| 4. | Реалізація ПЗ для моделювання | 24.04.2020 | Виконано |
| 5. | Розробка інтерфейсу | 26.04.2020 | Виконано |
| 6. | Тестування і виправлення помилок | 29.04.2020 | Виконано |
| 7. | Оформлення попереднього варіанту пояснювальної записки | 14.05.2020 | Виконано |
| 8. | Розширення функціоналу ПЗ | 17.05.2020 | Виконано |
| 9. | Повторне тестування | 18.05.2020 | Виконано |
| 10. | Подання ДП на попередній захист | 25.05.2020 | Виконано |
| 11. | Доповнення і подання ДП рецензенту | 08.06.2020 | Виконано |
| 12. | Подання ДП на основний захист | 09.06.2020 | Виконано |

Студент Олександр МІШИН

Керівник Володимир ШЕМАЄВ

# АНОТАЦІЯ

Мішин О.В. Застосунок моделювання цифрових фільтрів на основі .Net. КПІ ім. Ігоря Сікорського, Київ, 2020.

Проєкт містить 4 розділів, 60 сторінок, 7 рисунків, 14 таблиць та 17 посилань на літературні.

Ключові слова: цифровий сигнал, цифровий фільтр, передавальна функція, частотна характеристика, C#, WPF.

Об’єктом розробки є програма моделювання цифрових фільтрів. Мета розробки – створити додаток з інтуїтивно зрозумілим інтерфейсом, в якому можна буде розраховувати дію довільного фільтра на довільний сигнал. У дипломному проєкті розроблено систему, яка дозволяє завантажувати фільтри иа сигнали, наочно демонструвати їх дію і зберігати результуючий сигнал та частотні характеристики. Було проведено аналіз програм-аналогів, що дозволило прийняти ряд рішень щодо функціоналу програмного забезпечення. Було обрано технологію WPF, яка дозволяє розробити зручний і гнучкий інтерфейс для зручного зображення даних.

Отримана програма буде корисною для моделювання різного роду фільтрів, а також в навчальних і демонстраційних цілях. Функціонал системи буде розширюватись та вдосконалюватись.

# SUMMARY

Mishyn O.V. Digital filter modelling application based on .Net. Igor Sikorsky Kyiv Polytechnic Institute , Kyiv, 2020.

The project contains 4 sections, 60 pages, 7 figures, 14 tables and 17 references.

Keywords: digital signal, digital filter, transfer function, frequency response, C #, WPF.

The object of development is a program for modeling digital filters. The purpose of the development is to create an application with an intuitive interface that allows you to calculate the effect of any filter on any signal. In the diploma project system that allows you to load filters and signals, clearly demonstrate their result and save the resulting signal and frequency response was developed. An analysis of similar programs was performed, which allowed us to make a number of decisions about the functionality of the software. WPF technology was chosen because it allows us to develop a user-friendly and flexible interface for easy data display.

The resulting program will be useful for modeling various filters, as well as for educational and demonstration purposes. The functionality of the system will be expanded and improved.

| Поз. | Формат | Позначення | Найменування | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| 1 | | | Документація загальна | | | |
| 2 | | | Знову розроблена | | | |
| 3 | | | | | | |
| 4 | А4 | ІТ61.120БАК.004 ПЗ | Застосунок моделювання | 60 | | |
| 5 | | | цифрових фільтрів на основі | | | |
| 6 | | | .Net. | | | |
| 7 | | | Пояснювальна записка. | | | |
| 8 | А3 | ІТ61.120БАК.004 Д1 | Застосунок моделювання | 1 | | |
| 9 | | | цифрових фільтрів на основі | | | |
| 4 | | | .Net. | | | |
| 5 | | | Діаграма класів. | | | |
| 6 | А3 | ІТ61.120БАК.004 Д2 | Застосунок моделювання | 1 | | |
| 7 | | | цифрових фільтрів на основі | | | |
| 8 | | | .Net. | | | |
| 9 | | | Діаграма прецедентів. | | | |
| 10 | А3 | ІТ61.120БАК.004 Д3 | Застосунок моделювання | 1 | | |
| 11 | | | цифрових фільтрів на основі | | | |
| 12 | | | .Net. | | | |
| 13 | | | Блок-схема алгоритму ШПФ. | | | |
| 14 | А3 | ІТ61.120БАК.004 Д4 | Застосунок моделювання | 1 | | |
| 15 | | | цифрових фільтрів на основі | | | |
| 16 | | | .Net. | | | |
| 17 | | | Діаграма активності. | | | |

| | | | | | ІТ61.120БАК.004 ТП | | |
|---|---|---|---|---|---|---|---|
| Змн. | Лист | № докум. | Підпис | Дата | | | |
| Розроб. | | Мішин О.В. | | | Застосунок моделювання цифрових фільтрів на основі .Net. Відомість технічного проєкту | Літ. | Лист 1 / Листів 1 |
| Перевір. | | Шемаєв В.М. | | | | | |
| Н. Контр. | | | | | | КПІ ім. Ігоря Сікорського кафедра АУТС гр. ІТ-61 | |
| Затверд. | | Ролік О.І. | | | | | |

# Пояснювальна записка

## до дипломного проєкту

# на тему: «Застосунок моделювання цифрових фільтрів на основі .Net»

Київ – 2020

ЗМІСТ

# ВСТУП

Цифрові пристрої в наш час все більше витісняють аналогові. На це є багато причин. Цифрові дані можна копіювати, зберігати і передавати без втрат та накопичення похибок, чого неможливо позбутися повністю для аналогових сигналів. Через це проектування пристроїв, що дозволяють обробляти цифрові сигнали, є важливою складовою сучасної технології.

Цифрові фільтри є одним з типів таких пристроїв. Фільтри в загалому виконують наступну задачу: приймають певний сигнал на вхід, і повертають інший сигнал на виході. Хоча спочатку вони були лише аналогами аналогових фільтрів, що дозволяли відкидати від сигналів певні шуми і виділяти лише корисну частину, властивості цифрових сигналів дозволили узагальнити і розширити можливості цифрових фільтрів.

В даному дипломному проєкті буде розроблене програмне забезпечення, що дозволяє моделювати поведінку цифрових фільтрів. Для цього перш за все необхідно дослідити, як саме працюють цифрові фільтри і як можна побудувати їх модель. Це вимагає досконалого розуміння теоретичної бази.

При проектуванні важливим є дослідження програм-аналогів, які дозволяють виконувати схожі функції. Необхідно проаналізувати їх переваги та недоліки і розробити список вимог до програми, які виділять її серед аналогів.

Основною задачею такої програми є обчислити результат, який буде отримано, якщо на вхід певного фільтра подати певний сигнал. Програма також має дозволяти завантажувати фільтри та сигнали з файлів, та зображувати результуючий сигнал в графічній формі для зручного порівняння вхідного і вихідного сигналів. Для більш детального дослідження сигналів програма також має виконувати перетворення Фурьє, що дозволить дослідити вплив фільтра на частотні характеристики сигналів. Необхідно

дослідити, які саме характеристики зазвичай доводиться використовувати при розробці і експлуатації фільтрів.

З архітектурної точки зору програма має бути побудована з врахуванням загальноприйнятих домовленостей, що дозволять простіше тестувати, доповнювати та розширювати її. Передбачається повноцінне тестування як окремих компонент програми, так і програми в цілому при виконанні можливих дій користувача.

Важливою складовою проекту є розроблення технічної документації та керівництва користувача.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Теоретичні відомості

### 1.1.1 Цифрові сигнали

Для того, щоб більш детально розглянути переваги цифрових сигналів, слід розглянути в першу чергу аналогові сигнали. Аналоговий сигнал представляє собою деяку величину, яка неперервно змінюється з часом. Звичайно працюють з електричними сигналами, оскільки легко перевести інші види сигналів в електричний та навпаки. Для електричних сигналів змінною величиною звичайно є напруга на деякому елементі електричного кола. Максимальне(за модулем) значення напруги, що може дати сигнал, називається амплітудою сигналу. З математичної точки зору аналоговий сигнал є неперервною функцією напруги від часу, і його можна аналізувати як функцію: проаналізувати його максимуми/мінімуми, періодичність, знайти швидкість його зміни(похідну) та результат накопичення сигналу(інтеграл). Існують аналогові пристрої, які здатні розрахувати ці параметри для вхідного сигналу.

Цифровий сигнал представляє собою послідовність дискретних значень. Замість неперервної функції від часу такий сигнал являється послідовністю імпульсів фіксованої ширини[1]. В природі цифрові сигнали не існують, але можна перетворити аналоговий сигнал на цифровий, виконавши процедури дискретизації та квантування.

Дискретизація полягає в вимірюванні сигналу в певні моменти часу. Надалі замість неперервної функції сигнал представляється як послідовність виміряних значень. Інтервал між проведенням вимірів називається періодом дискретизації, а обернена йому величина – частотою дискретизації. При проведенні цієї процедури очевидно втрачається деяка інформація про сигнал, а саме – будь-яка інформація про поведінку сигналів між вимірами. Збільшення частоти дискретизації частково зменшує цю проблему, але не

розв'язує її повністю, крім цього збільшення частоти дискретизації в X разів збільшує об'єм пам'яті для збереження в стільки ж разів[2].

Квантування полягає в розбиванні інтервалу можливих значень на скінченну кількість рівнів. Надалі значення сигналу в будь-який момент часу замінюється на значення, округлене до найближчого рівня. При цьому втрачається інформація про зміну сигналу в межах одного рівня, але це також можна покращити збільшенням кількості рівнів. Після квантування, кожний рівень можна закодувати числом. В комп'ютерах числа записуються в двійковому вигляді, і важливою характеристикою квантування є розрядність – кількість біт, необхідна для кодування рівня квантування. Подвоєння кількості рівнів призводить до збільшення розрядності на 1[3].

Хоча проведення цих операцій призводить до втрат, які неможливо повністю вилучити, одержаний цифровий сигнал має свої переваги. Його легко представити в двійковому вигляді, після чого будь-яка похибка копіювання чи передачі, яка є недостатньо сильної щоб змінити 0 на 1 чи навпаки, абсолютно ніяк не впливає на сигнал, і результатом передачі чи копіювання буде його точна копія, на відміну від аналогового сигналу, копіювання якого завжди є неточним і привносить похибки, які можуть накопичитись з часом. Навіть просто тривале збереження аналогового сигналу призводить до накопичення незначних похибок з часом, що не є проблемою для цифрового сигналу.

Ці переваги призвели до того, що в наш час аналогові пристрої майже не використовуються, і вся сучасна обчислювальна техніка є цифровою.

### 1.1.2 Комплексні числа

Хоча всі реально існуючі величини є дійсними, часто буває корисно розглядати комплексні числа в проміжних обчисленнях. Немає необхідності

повністю описувати всі можливі дії над комплексними числами, вкажемо лише їх основні властивості[4]:

– комплексне число – число виду a+bi, де a, b – дійсні, а i – уявна одиниця: i*i=-1;

– комплексні числа додаються і віднімаються почленно;

– комплексні числа множаться почленно з врахуванням того, що i*i=-1;

– формула Ейлера: $\cos(\varphi) + i * \sin(\varphi) = e^{i\varphi}$, з якої випливає експоненційна форма запису комплексних чисел: $a + bi = r(\cos(\varphi) + i * \sin(\varphi)) = re^{i\varphi}$, де $r = \sqrt{a^2 + b^2}$ – модуль числа, а $\varphi$ – аргумент;

– множення і ділення чисел в експоненційній формі відбувається як множення/ділення модулів і додавання/віднімання аргументів.

### 1.1.3 Цифрові фільтри

В даній роботі розглядаються фільтри з одним входом і виходом. Ці фільтри приймають на вхід один сигнал і видають інший сигнал на виході. В будь-який момент значення на виході може залежати не тільки від поточного значення на вході, а і від попередніх значень як вхідного, так і вихідного сигналів. Більша частина фільтрів є лінійними і стаціонарними.

Лінійність означає, що лінійна комбінація входів переходить в лінійну комбінацію виходів, а саме: множення сигналу на число(однакове для всіх значень) призводить до того, що результат також множиться на те ж число, а також що сума сигналів переходить в суму їх результатів. Наприклад, фільтр, який підсилює сигнал вдвічі, є лінійним, а фільтр, який підсилює його на константу, не є лінійним.

Стаціонарність означає, що результуюче значення не залежить явно від часу. Це значить, що результуюче значення не залежить від номеру значення, а значить що весь сигнал можна зсунути в часі і результат буде незмінним, лише так само зсунутим в часі. Наприклад, фільтр, що усереднює 2 останні

вхідні значення, є стаціонарним, а фільтр, що усереднює поточне вхідне значення з першим, не є стаціонарним[5].

Лінійні стаціонарні фільтри простіше проектувати і розробляти, тому вони є більш популярними, і в даній роботі надалі будуть розглядатися лише такі фільтри.

Також ми будемо розглядати лише фільтри, що можуть працювати в реальному часі. Такі фільтри мають генерувати вихідний сигнал лише за тими значеннями вихідного і вхідного сигналів, які вже були отримані. Такі фільтри називаються каузальними. Наприклад, фільтр, який би сортував значення вхідного сигналу і виводив їх в порядку зростання, не був би стаціонарним, і не міг би працювати в реальному часі, оскільки для того, щоб отримати перше значення вихідного сигналу, треба знати всі майбутні значення вхідного.

З врахуванням цих властивостей ми можемо сказати, що результуюче значення в момент i, $y_i$, є лінійною функцією наступних величин: значення вхідного сигналу в момент i ($x_i$), значення вхідного сигналу в попередні декілька моментів($x_{i-1}$, $x_{i-2}$,...,$x_{i-n}$) і значення результату в попередні декілька моментів часу($y_{i-1}$, $y_{i-2}$,...,$y_{i-m,}$):

$$y_i = \sum_{j=0}^{n} b_j x_{i-j} - \sum_{j=1}^{m} a_j y_{i-j} \qquad (1)$$

Тут знак «-» перед другою сумою обрано для зручності запису подальших формул. Коефіцієнти $b_0..b_n$ і $a_1..a_m$ разом повністю задають фільтр, дозволяючи розрахувати значення вихідного сигналу, використовуючи вже відомі попередні значення вихідного сигналу, та поточне і попередні значення вхідного.

Було б корисно побудувати певний математичний апарат, що дозволив би звести обчислення результату до множення на певне значення. Можна

побудувати функцію, яка буде рівна відношенню y(t)/x(t), але це значення буде просто числом, яке, до того ж, залежить не тільки від самого фільтру, а й від попередніх значень сигналів. Замість цього ми використаємо засіб, який в математиці відомий як z-перетворення. Для нашої задачі зміст перетворення не є важливим, лише його властивості.

Для послідовності чисел a(n) її z-перетворення це така функція комплексної змінної A(z), що

$$A(z) = \sum_{n=1}^{\infty} a_n z^{-n} \qquad (2)$$

Аналогічно, обернене z-перетворення функції A(z) – послідовність a(n) така, що її z-перетворення дасть цю функцію.

Ми використаємо наступні властивості z-перетворення:

Обидва перетворення є однозначними, тому якщо послідовності рівні, їх перетворення-рівні і навпаки. Однозначність прямого перетворення очевидна, оскільки сума є однозначною. Однозначність оберненого перетворення випливає з того, що результуючі суми мають давати однакові результати для всіх z.

Обидва перетворення є лінійними(множення послідовності на число призводить до множення всіх доданків суми на це число, а значить і результату; заміна в означенні $a_n$ на $a_n+b_n$ призведе до розкладу суми на 2 суми, за a і b відповідно).

Перетворення b(n)=a(n-1) дає

$$B(z) = \sum_{n=1}^{\infty} b_n z^{-n} = \sum_{n=1}^{\infty} a_{n-1} z^{-n} = \sum_{n=1}^{\infty} a_{n-1} z^{-(n-1)} z^{-1} = z^{-1} A(z) \qquad (3)$$

Це значить, що зсув послідовності на 1 назад призводить до множення перетворення на $z^{-1}$[6].

Застосуємо це для отриманої y(i) в (1), виконавши перетворення лівої і правої частини. Нехай x(i)->X(z), y(i)->Y(z), тоді, з врахуванням лінійності і правила зсуву:

$$Y(z) = \sum_{j=0}^{n} b_j X(z) z^{-j} - \sum_{j=1}^{m} a_j Y(z) z^{-j} \quad (4)$$

$$Y(z) + \sum_{j=1}^{m} a_j Y(z) z^{-j} = \sum_{j=0}^{n} b_j X(z) z^{-j} \quad (5)$$

Оскільки в першій сумі j починається з 1, можна ввести $a_0$=1, тоді

$$Y(z) * \sum_{j=0}^{m} a_j z^{-j} = X(z) * \sum_{j=0}^{n} b_j z^{-j} \quad (6)$$

Тепер можна ввести передавальну функцію H(z)=Y(z)/X(z)

$$H(z) = \frac{\sum_{j=0}^{n} b_j z^{-j}}{\sum_{j=0}^{m} a_j z^{-j}} \quad (7)$$

В такому вигляді вона залежить лише від коефіцієнтів фільтра, а не від самих сигналів[7]. З використанням цієї функції можна виконати наступні операції:

– послідовне виконання спочатку фільтра $H_1$, а потім $H_2$ можна замінити виконанням фільтра $H=H_1*H_2$, і навпаки;

– паралельне виконання над одним сигналом фільтрів $H_1$ і $H_2$ і додавання результатів можна замінити виконанням фільтра $H=H_1+H_2$, і навпаки.

1.1.4 Гармонічні сигнали і перетворення Фурьє

Гармонічними сигналами є сигнали виду

$$g(t) = Acos(\omega t + \varphi) \quad (8)$$

або, в комплексній формі,

$$g(t) = Ae^{i(\omega t+\varphi)} \quad (9)$$

(дійсна форма отримується з комплексної взяттям дійсної частини). Тут A – амплітуда сигналу, $\omega$ – частота, а $\varphi$ – його початкова фаза[8]. Для цифрового сигналу з періодом дискретизації T t=nT, і

$$g(n) = Ae^{i(kn+\varphi)} \quad (10)$$

де $k=\omega*T$ – константа, що називається індексом частоти. Можна довести, що будь-який сигнал можна розбити на лінійну комбінацію гармонічних сигналів(іноді нескінченну). Оскільки ми розглядаємо лінійні фільтри, можна проаналізувати, як фільтр, заданий коефіцієнтами, реагує на гармонічні вхідні сигнали.

Враховуючи лінійність, ми можемо розглянути сигнал $e^{ikn}$, оскільки загальний вигляд можна отримати, домноживши цей сигнал на $Ae^{i\varphi}$, що, з врахуванням лінійності, призведе до такого ж домноження результату.

Підстановка такого сигналу в фільтр дає наступні результати:

– Результуючий сигнал теж є гармонічним, і має таку ж частоту.

– Амплідуду результуючого сигналу можна знайти з передавальної функції: A(k)=|H($e^{ik}$)|

– Початкову фазу результуючого сигналу можна знайти з передавальної функції: f(k)=arg(H($e^{ik}$))

Виведення цих властивостей не є важливим для даної роботи і буде опущено[7].

Такі властивості гармонічних сигналів призводять до того, що фільтр досить часто можна охарактеризувати як пристрій, який змінює амплітуду і фазу кожної гармонічної компоненти в залежності від її частоти. Через це залежності зміни амплітуди і фази від частоти є важливими характеристиками фільтра і називаються відповідно АЧХ(амплітудно-частотна характеристика) і ФЧХ(фазово-частотна характеристика).

Досить поширеними видами фільтрів є фільтри, задача яких – пропускати чи не пропускати певні частоти. Це фактично означає що нам потрібен фільтр з деякою наперед заданою АЧХ. Як правило, ідеально відфільтрувати певні частоти і ніяк не змінити компоненти потрібних нам частот неможливо, тому в залежності від того, яка саме задача розв'язується фільтром, використовують різні варіанти фільтрів.

Для аналізу частотних складових сигналів використовується перетворення Фурьє. Воно має наступний вид:

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ix\omega} dx \qquad (11)$$

Зміст цього перетворення наступний: $\hat{f}(\omega)$ вказує «відносну амплітуду» гармонічної складової з частотою $\omega$ в сигналі $f(x)$. Воно є лінійним, а розклад гармонічного сигналу дає ненульове(нескінченно велике) значення для $\omega$ рівного частоті сигналу.

Для дискретних сигналів використовується так зване дискретне перетворення Фурьє, яке має наступний вид:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{2\pi i}{N} kn} \quad (12)$$

Його зміст - аналогічний, $X(k)$ – амплітуда гармонічної складової з індексом частоти k. Ця амплітуда є комплексною, а тому виражає як власне амплітуду(модуль $X(k)$), так і фазу(аргумент $X(k)$)[9].

### 1.1.5 Швидке перетворення Фурьє

В даній роботі перетворення Фурьє виконується не за визначенням, а з використанням алгоритму «швидке перетворення Фурьє». Воно є значно швидшим за звичайне, але вимагає, щоб кількість елементів була степенем двійки.

Для цього спочатку замінимо формулу перетворення наступним чином:

$$e^{-\frac{2\pi i}{N} kn} = a^{\frac{kn}{N}}; X(k) = \sum_{n=0}^{N-1} x(n) a^{\frac{kn}{N}} \quad (13)$$

Тепер розіб’ємо суму на 2 частини: за парними та за непарними елементами:

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)\, a^{\frac{2kn}{N}} + \sum_{n=0}^{N/2-1} x(2n+1)\, a^{\frac{k(2n+1)}{N}} \quad (14)$$

Далі перетворимо степені $a$ так, щоб в знаменнику було $N/2$:

$$a^{\frac{2kn}{N}} = a^{\frac{kn}{\frac{N}{2}}};\ a^{\frac{k(2n+1)}{N}} = a^{\frac{k}{N}} * a^{\frac{kn}{\frac{N}{2}}} \tag{15}$$

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)\, a^{\frac{kn}{\frac{N}{2}}} + a^{\frac{k}{N}} \sum_{n=0}^{N/2-1} x(2n+1)\, a^{\frac{kn}{\frac{N}{2}}} \tag{16}$$

Скориставшись тим, що $a^{\frac{k}{N}} = e^{-\frac{2\pi i}{N}k}$ та іншими властивостями комплексних чисел, після деяких громіздких обчислень, які тут буде опущено, отримуємо результат:

$$X(k) = E_k + e^{-\frac{2\pi i}{N}k} O_k;\ X(k+N/2) = E_k - e^{-\frac{2\pi i}{N}k} O_k \tag{17}$$

Де $E_k, O_k$ – відповідно перетворення Фурьє для підмасивів парних та непарних елементів[10].

Оскільки кожна пара $E_k, O_k$ використовується двічі, це призводить до зменшення кількості операцій множення: для розрахунку за визначенням необхідно $N^2$ операцій(N значень, кожне з яких обчислюється за N операцій), а для розрахунку за отриманими формулами, якщо E і O розраховуються за визначенням, $2 * \left(\frac{N}{2}\right)^2 = \frac{N^2}{2}$. Якщо ж N є степенем двійки, то кожний підмасив можна так само перетворити цим ж методом, отримавши рекурсивний алгоритм складністю O(N*logN).

Отримані формули нескладно записати в алгоритмічному вигляді для автоматичного розрахунку. В даній роботі перетворення Фурьє виконується саме цим способом, оскільки звичайне ДПФ працює значно повільніше для

довгих сигналів. Блок-схема даного алгоритму наведена на кресленику ІТ61.120БАК.004 Д3.

Отримані значення, як і для звичайного перетворення, є комплексними. Через це слід передбачити розбиття цих чисел на складові: модуль і аргумент.

### 1.1.6 Частотні характеристики фільтра

Як вже було вказано вище, важливими характеристиками фільтра є АЧХ(амплітудно-частотна характеристика) і ФЧХ(фазово-частотна характеристика), які обчислюються підстановкою в передавальну функцію $e^{ik}$ . Для зменшення кількості операцій обчислення виконується за наступним алгоритмом:

– $z^0 = 1$;

– на k-му кроці ми домножуємо $z^{-k}$ на відповідні коефіцієнти чисельника і знаменника і додаємо до їх поточних значень;

– домножуємо $z^{-k}$ на $e^{-ik}$, отримавши $z^{-(k+1)}$;

– після виконання останнього кроку ділимо результуючі значення чисельника і знаменника;

– отримане комплексне число розбивається на модуль(що дає значення АЧХ) і аргумент(що дає значення ФЧХ).

Цей алгоритм дозволяє мінімізувати кількість обчислень для розрахунку характеристик. З врахуванням того, що обидві характеристики вимагають обчислення одного й того ж самого числа, а його обчислення займає більшу частину операцій, вважається доцільним обчислювати одночасно обидві характеристики.

## 1.2 Аналіз програм-аналогів

На даний момент популярними засобами, що використовуються для моделювання і проектування цифрових фільтрів, є математичні пакети, такі як Matlab. Оскільки інші пакети мають аналогічний функціонал, ми будемо розглядати лише Matlab, як найбільш популярний з них.

Крім цього, існують спеціалізовані програми, розроблені саме для задач роботи з фільтрами. Серед таких програм ми будемо розглядати програми FIWIZ та FilterLab

### 1.2.1 Matlab

Matlab – популярний математичний пакет, який використовується для проектування, математичного моделювання, розвязання задач і аналізу даних. В контексті розглянутої задачі його можна використовувати наступним чином:

– побудувати сигнал за певними формулами чи створити випадкові шуми різного роду;

– застосувати певну передавальну функцію до сигналу (приклад наведено на рис. 1.1);

– графічно зобразити характеристики фільтрів та сигналів;

– логічно приєднати фільтр до інших змодельованих приладів(джерела сигналів тощо)[11].

Хоча цей пакет і здатний розвязувати практично всі задачі, повязані з моделюванням фільтрів, у нього є свої недоліки. Основний недолік – ціна, Matlab не є безкоштовною програмою, і, у випадку якщо передбачається комерційне використання, його ціна може становити більше 2 тисяч долларів, чи більше ніж 900 долларів за рік ліцензії[12].

```
y = filter(b,a,x);

plot(t,x)
hold on
plot(t,y)
legend('Input Data','Filtered Data')
```

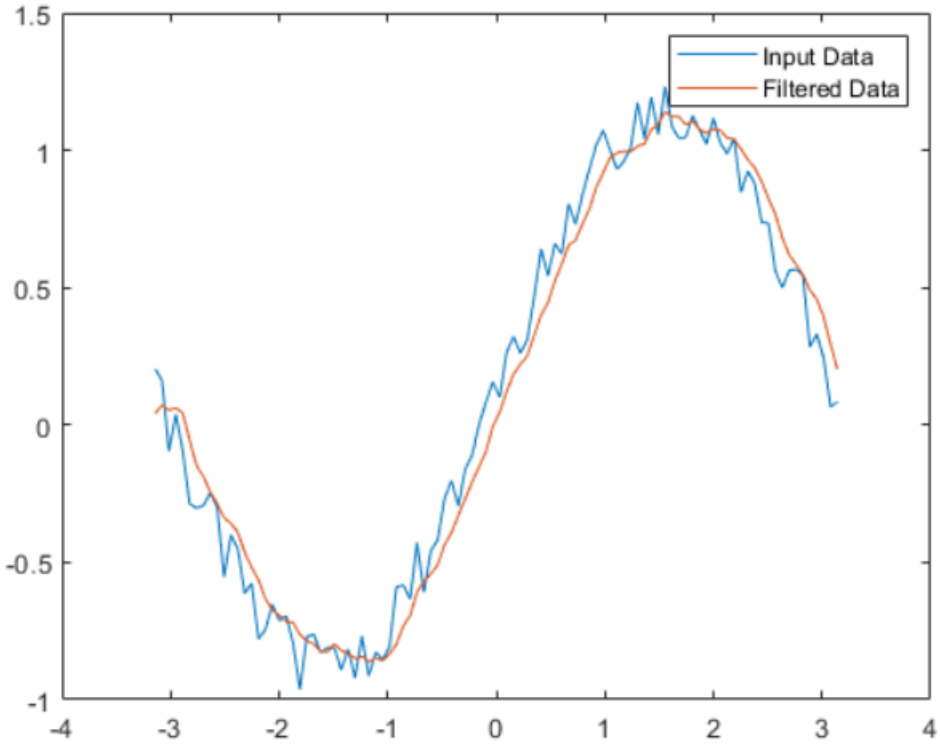


Рисунок 1.1 – Код Matlab, що застосовує фільтр до сигналу, і результат його дії, наведений графічно

Крім цього, синтаксис Matlab, хоч і не є складним, вимагає часу на засвоєння. Незважаючи на це, якщо, крім моделювання фільтрів, перед користувачем стоять й інші задачі, повязані з математичним моделюванням та іншим, використання цієї програми може бути більш оптимальним, ніж використання декількох окремих спеціалізованих програм.

### 1.2.2 FIWIZ

FIWIZ – спеціалізована програма, яка дозволяє проектувати фільтри декількох обраних типів в залежності від вимог користувача. Серед таких типів – рекурсивні фільтри Баттерворда, Чебишева та еліптичні; нерекурсивні фільтри та ін. Програма дозволяє згенерувати фільтр в залежності від вимог користувача, таких як обмеження на АЧХ та ФЧХ, як зображено на рисунку 1.2, а також дозволяє записати фільтр в файл, який може відкрити Matlab[13].

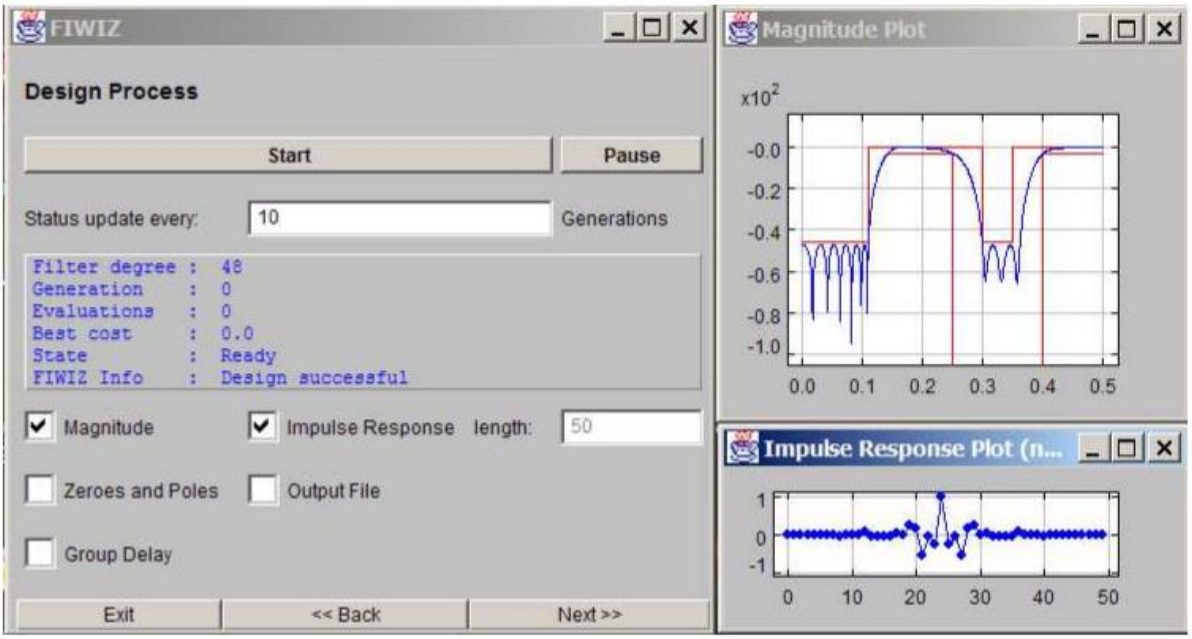


**Figure 5.5-2: Example for linear phase FIR design with maxflat constraint in multiple passbands.**

Рисунок 1.2 – Один з етапів проектування фільтру в FIWIZ. Наведені графіки вимог до АЧХ і результуючої АЧХ, а також графік фазової характеристики

Ця програма є корисною для проектування фільтрів низьких та високих частот, а також смугових і загороджуючих. Ще однією перевагою є те, що ця програма є безкоштовною. Основним недоліком цієї програми є

неможливість змоделювати довільний фільтр чи дослідити його дію на довільний сигнал.

### 1.2.3 FilterLab

FilterLab – ще одна спеціалізована програма, яка призначена саме для проектування фільтрів. Присутня невелика кількість варіантів, але система діалогів дозволяє покроково ввести необхідні параметри і обрати варіанти, які найкраще відповідають цим вимогам[14]. Вигляд вікна вибору типу фільтру наведено на рис. 1.3.

Крім цього є можливість переглянути фізичну схему побудованого фільтру, і параметри елементів, необхідних для її побудови, як це зображено

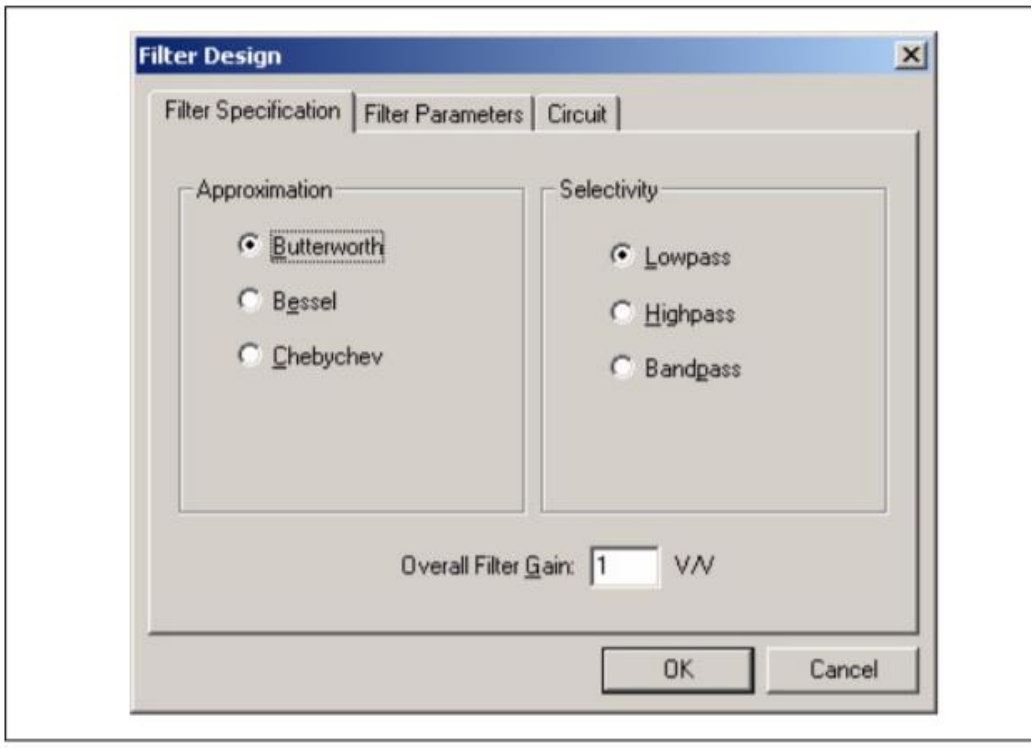


***FIGURE 1-1:*** *Filter Specification Tab*

Рисунок 1.3 – Вибір моделі фільтру та його методу пропускання частот

на рис. 1.4.

Як і попередня програма, FilterLab не дозволяє змоделювати довільний фільтр, а вбудованих варіантів ще менше, ніж у FIWIZ, вона найкраще підійде для фізичного проектування фільтрів стандартних типів.

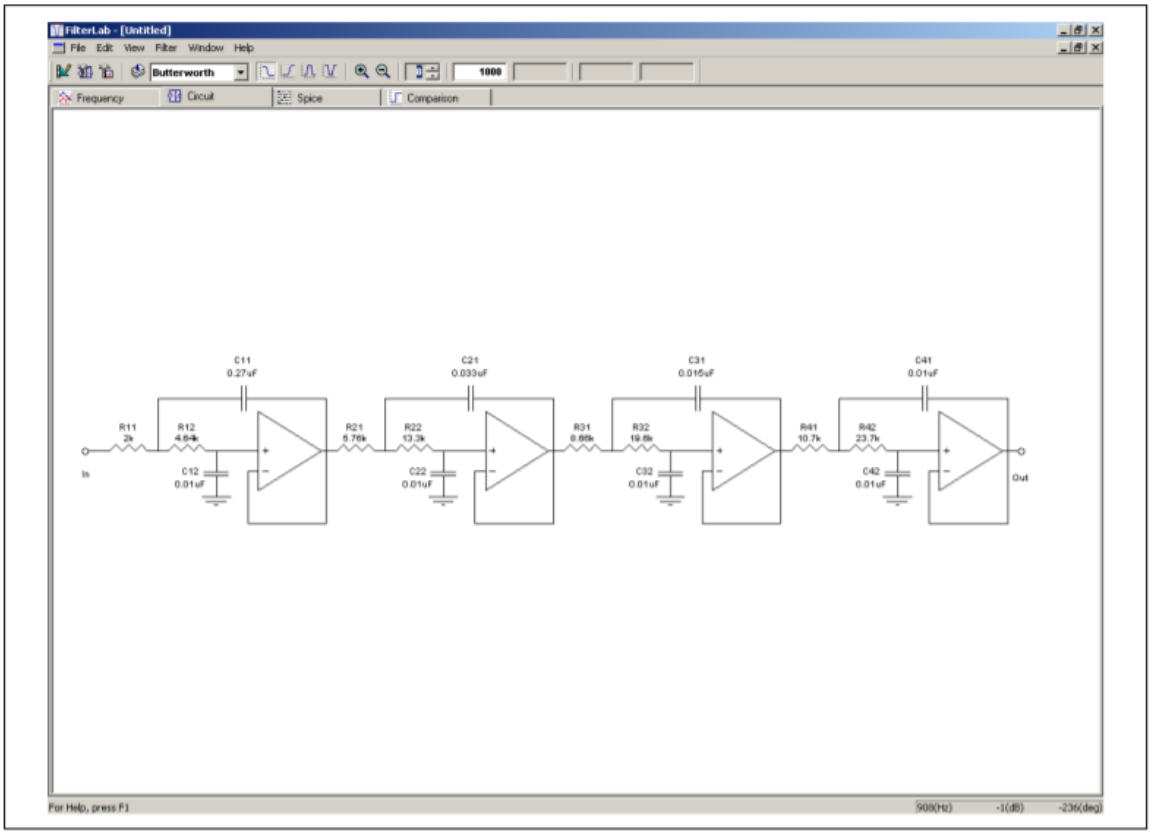


***FIGURE 5-2:*** *Circuit View*

Рисунок 1.4 – Фізична схема фільтру, побудована в FilterLab

## 1.3 Порівняння обраних аналогів

В таблиці 1.1 наведено результати порівняння обраних програм-аналогів. З таблиці видно, що, для того, щоб наша програма виділялася серед аналогів, слід розробити зручний інтерфейс і дозволити моделювати довільний фільтр, а от побудова спеціалізованих фільтрів не є необхідною вимогою

Таблиця 1.1 – Порівняння програм-аналогів

| Назва ознаки, що порівнюється | Matlab | FIWIZ | FilterLab |
|---|---|---|---|
| Ціна | - | + | + |
| Моделювання довільного фільтру | + | - | - |
| Назва ознаки, що порівнюється | Matlab | FIWIZ | FilterLab |
| Побудова фільтру за призначенням | + | + | + |
| Фізичне проектування схеми фільтру | +/- | - | + |
| Зручність використання непідготовленим користувачем | +/- | - | +/- |
| Збереження результатів в зовнішні файли | + | - | - |

## 1.3 Розробка вимог до програмного забезпечення

Основною вимогою до програми є можливість моделювати довільний фільтр. Моделювання включає в себе наступні складові:

— завантаження довільного фільтру;

— передача на вхід фільтру довільного сигналу;

— отримання результату дії фільтру на сигнал;

— отримання частотних характеристик фільтру(АЧХ, ФЧХ);

— отримання інших характеристик фільтру(порядок, рекурсивність, стійкість);

— отримання характеристик вхідного і вихідного сигналів(розподіл за частотами).

Крім цього, для зручності роботи вимагаються наступні можливості:

– графічне зображення характеристик сигналів та фільтрів;

– графічне порівняння сигналів;

– збереження результуючого сигналу для подальшого використання (наприклад для подачі його на вхід іншого фільтру);

– вбудована база стандартних фільтрів і сигналів, яку можна розширювати;

– графічне порівняння характеристик декількох фільтрів;

– графічне порівняння вихідних сигналів декількох фільтрів.

До способу збереження даних вимоги відсутні, але вимагається щоб розширення можливостей збереження даних за необхідності було максимально простим.

Інтерфейс програми має бути інтуїтивно зрозумілим.

## 1.4 Висновки до розділу

Ми дослідили теоретичну частину моделювання фільтрів і розглянули програми, які дозволяють розв’язувати задачу моделювання. Основні результати, що ми отримали, це:

– Фільтр можна змоделювати передавальною функцією, а саме коефіцієнтами в чисельнику і знаменнику. Побудова результуючого значення зводиться до множення цих коефіцієнтів на поточне і попередні значення сигналів і обчислення суми.

– Для отримання більш детальних відомостей про фільтри та сигнали в програмі слід реалізувати роботу з комплексними числами і алгоритм перетворення Фурьє.

– Спеціалізовані програми-аналоги непогано розв’язують задачу генерації фільтра за відомим призначенням фільтра, але зазвичай не дозволяють проаналізувати довільний фільтр. Потужні математичні пакети можуть розв’язати цю задачу, але вони є більш складними для використання, і є далеко не безкоштовними.

Було побудувано порівняльну таблицю обраних програм-аналогів для більш зручного порівняння їх переваг та недоліків.

З використанням отриманої таблиці, ми побудували вимоги до програмного забезпечення. Крім основних вимог, що дозволяють виконувати власне моделювання, важливими є також вимоги до зручності користування програмою.

# 2 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1 Структура програми

При розробленні програми будемо дотримуватись паттерну MVVW(model-view-viewmodel).

Model – власне обчислювальна складова.

View – інтерфейс користувача та прив'язана до нього користувацька логіка

ViewModel – проміжна складова, яка переводить дані, отримані від моделі, в зрозумілий для View формат, а також отримує команди користувача і передає їх в модель, викликаючи її методи.

З точки зору View, ViewModel це абстракція моделі, View передає команди користувача в ViewModel і отримує з неї дані. Model отримує від ViewModel команди користувача і передає туди дані[15].

В даному проекті взаємодія між цими складовими виконується наступним чином:

View передає на ViewModel команди інтерфейсу(обрати фільтр з варіантів, обрати сигнал з варіантів, завантажити новий фільтр з файлу, завантажити новий сигнал з файлу)

View отримує від ViewModel дані(дані про обраний фільтр та сигнал, масиви точок для графіків)

Model отримує від ViewModel запит перерахувати результат з використанням обраного фільтра та сигналу

Model повертає в ViewModel результуючий сигнал, розраховані частотні характеристики та результати перетворення Фурьє сигналів.

## 2.2 Опис класів

На кресленику ІТ61.120БАК.004 Д1 наведено детальну схему класів і зв’язків між ними, тут ми обмежимось коротким описом класів окремих складових.

### 2.2.1 Класи Model

До Model належать класи DFMCoefList, DFMFilter, DFMSignal і FastFourierTransform

DFMFilter є класом, що відповідає за фільтри, він зберігає 2 списки коефіцієнтів передавальної функції. Саме в ньому відбувається обчислення вихідного сигналу за вхідним, а також обчислюються характеристики фільтру. Крім цього, там зберігається назва фільтру, за якою його можна ідентифікувати.

DFMCoefList є допоміжним класом для DFMFilter, в ньому зберігається список коефіцієнтів для чисельника чи знаменника передавальної функції, а також наявні методи обчислення множення цих коефіцієнтів на відповідні значення сигналу.

DFMSignal є класом, що відповідає за сигнали, в ньому сигнал зберігається як послідовність цілих чисел. Наявні методи роботи з сигналами, такі як додавання, множення на число і зсув. Як і фільтри, сигнали можуть мати назву для ідентифікації.

FastFourierTransform відповідає за розрахунок частотних характеристик сигналів. В його методі виконується швидке перетворення Фурьє, і виводиться результат як послідовність комплексних чисел.

### 2.2.2 Класи ViewModel

До ViewModel належить лише клас ViewModel.

Обєкт цього класу зберігає наступні дані:

- обраний фільтр;
- обраний сигнал;
- результуючий сигнал;
- дані про фільтр;
- дані про сигнали;
- масиви точок для графіків.

При зміні обраного фільтру чи сигналу виконується перерахунок решти даних, при цьому викликається подія для View.

### 2.2.3 Класи View

До View належить лише клас MainWindow

В MainWindow є методи обробки натискання на кнопки чи обирання елементів зі списків. Ці методи викликають зміни в ViewModel, якщо вони відповідають за зміну сигналу чи фільтру. Крім цього, є можливість переключити режими графіків чи змінити їх розмір. Такі зміни викликають метод перерисування, але при цьому беруться вже готові дані з ViewModel, нічого при цьому не перераховується. Це дозволяє уникнути постійного перерозрахунку одних і тих самих даних при простому розтягненні вікна.

### 2.2.4 Інші класи

Створено допоміжний клас ComplexDouble, що дозволяє проводити обчислення з комплексними числами. Це є корисним при роботі з швидким перетворенням Фурьє, а також при розрахунку АЧХ та ФЧХ фільтра.

Для роботи з зовнішніми джерелами даних створено інтерфейси ISignalController і IFilterController, та їх реалізації для роботи з файлами: SignalFileController і FilterFileController. Інтерфейси написано таким чином,

щоб в подальшому можна було створити інші контролери, що працюють з іншими джерелами даних.

Класи …FileController включають в себе логіку зчитування з файлів і запису в файл. В них вказані правила зчитування/запису, а також виконується перевірка на правильність. За необхідності правила та можливості створення файлів можна буде розширити через ці класи, при цьому вимагається сумісність старих фільтрів/сигналів з новими методами зчитування.

## 2.3 Зовнішнє збереження даних

В даному проекті збереження даних про фільтри та сигнали відбувається в текстових файлах.

### 2.3.1 Файли фільтрів

Файл фільтру має наступний формат:

Filter <назва>

<коефіцієнти чисельника передавальної функції>

<коефіцієнти знаменника передавальної функції>

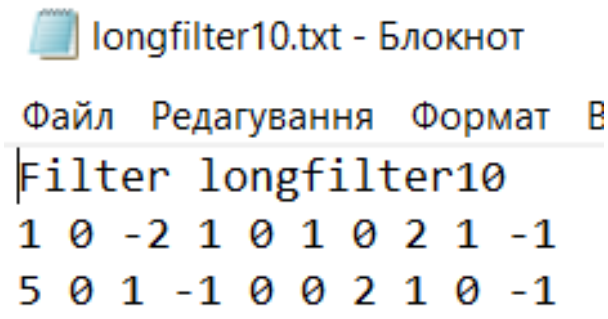


Рисунок 2.1- Приклад файлу фільтру

Коефіцієнти вказуються в порядку збільшення степеня при z^-1, і можуть бути дійсними.

Перший коефіцієнт знаменника має бути ненульовим.

Приклад файлу наведено на рис. 2.1.

### 2.3.2 Файли сигналів

Файл сигналу має наступний формат

Signal <назва>

<модифікатор продовження><перелік значень>

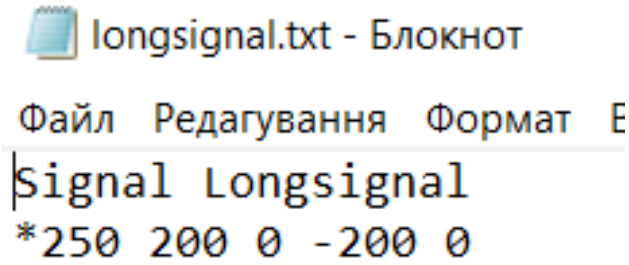


Рисунок 2.2 – Приклад файлу сигналу

Модифікатор продовження вказує, скільки разів повинен повторюватись цей сигнал. Він записується як *<число>, і означає "повторити цю послідовність стільки раз". Це дозволяє користувачу простіше створювати власні довгі послідовності сигналів, але сама програма при збереженні сигналів просто зберігає всю послідовність.

Приклад файлу з використанням модифікатора наведено на рис. 2.2.

### 2.3.3 Файли частотних характеристик

Такі файли створюються програмою і не читаються нею. Формат файлу:

<назва елементу> <назва характеристики>

<перелік елементів>

## 2.4 Мова програмування, середовище і технології

### 2.4.1 Мова програмування

Мовою програмування для цього проекту була обрана мова C#. Причини такого вибору наступні: для створення даної програми зручно використати об’єктно-орієнтовану мову програмування, оскільки вже на етапі проектування легко розбити програму на складові з певними функціями і способами взаємодії. Крім цього C# є строго типізованою мовою програмування, що робить програму більш стійкою до runtime-помилок. Іншою мовою, що задовольняє цим вимогам, є Java, і вибір між цими мовами визначається здебільшого вибором цільової платформи та особливостями синтаксису мов.

### 2.4.2 Середовище розробки

Хоча розробка програмного забезпечення можлива в будь-якому текстовому редакторі, використання спеціалізованого середовища розробки значно спрощує цей процес. Для мови C# найбільш поширеним є середовище розробки Microsoft Visual Studio. Воно дозволяє повноцінно використовувати можливості мови C#, дозволяючи автозаповнення і підсвітку помилок. Також в цьому середовищі присутня підтримка різного роду методів тестування. Саме воно, а саме версія Community 2019, було обране для розробки проекту.

### 2.4.3 Платформа .NET Framework і технологія WPF

.NET – платформа, що дозволяє використовувати можливості CLR(Common Language Runtime) незалежно від використовуваної мови програмування. Використання C# передбачає використання .NET. Хоча в даній роботі

розробка виконується однією мовою, використання засобів CLR робить розробку простішою[16].

WPF(Windows Presentation Foundation) – набір засобів для створення програм, що включає в себе різного роду засоби для розробки користувацьких інтерфейсів. Серед цих засобів-зручна організація адаптивного розподілу місця між елементами, механізми прив'язки даних, XAML-розмітка і робота з графікою. Оскільки зручність роботи з користувацьким інтерфейсом і наочне зображення результатів є вимогами до програми, WPF технологія є корисною при розробці[17].

### 2.4.4 Система контролю версій Git

Система контролю версій є необхідною складовою розробки програмного забезпечення в сучасному світі. Для розробки цієї програми була обрана система контролю версій Git. Основною перевагою цієї системи є її популярність: більшість розробників програмного забезпечення використовують Git, тому, якщо виникає потреба додати розробників до проекту, це легше зробити, ніж при використанні інших систем. Крім цього, середовище розробки Visual Studio підтримує цю систему контролю версій.

## 2.5 Висновки до розділу

В даному розділі ми побудували структуру програми і обрали засоби і технології для розробки. Було прийняте рішення будувати рішення за обєктно-орієнтованою моделлю, обрана архітектура MVVM, побудовані базові класи для компонент архітектури, і способи взаємодії між ними.

Було обрано мову програмування C#, середовище розробки MS Visual Studio 2019, технології .NET та WPF, та систему контролю версій Git.

# 3 АНАЛІЗ ЯКОСТІ І ТЕСТУВАННЯ

## 3.1 Опис об'єктів для тестування

Існують різні методи тестування. В даній роботі були використані юніт-тести, тести сценаріів і тести продуктивності.

Юніт-тести тестують якомога менші елементи програми, зазвичай це окремі методи.

Юніт-тестами тестуються наступні елементи:

- DFMCoefList;
- DFMFilter;
- ComplexDouble;
- DFMSignal.

Тести сценаріїв передбачають виконання певного користувацького сценарію.

В даній роботі тестуються наступні сценарії:

- застосування фільтру та сигналу за замовченням і одержання результату;
- завантаження фільтру з файлу;
- завантаження сигналу з файлу;
- збереження даних в файл;
- збереження сигналу в файл і подальше завантаження цього сигналу.

Тестами продуктивності виявляються обмеження на обсяг вхідних даних.

В даній роботі тестами виявляються обмеження на:

- розмір сигналу;
- порядок фільтру.

3.2 Перелік проведених тестів

3.2.1 Юніт-тести

В таблиці 3.1 наведено завершальний перелік юніт-тестів.

Таблиця 3.1 – Юніт-тести

| Метод, що тестується | Опис тесту і очікуваний результат | Результат тесту |
|---|---|---|
| DFMCoefList. MultiplyBySignal | Результат – число, що відповідає результату почленного множення коефіцієнтів на значення сигналу | Пройдено |
| DFMFilter. GenerateOutput | Результуючий сигнал має довжину, що відповідає вказаній, і значення сигналу відповідають значенням вхідного сигналу та коефіцієнтам фільтру | Пройдено |
| DFMFilter.Order | Порядок фільтру має відповідати номеру найбільшого ненульового коефіцієнта | Пройдено |
| ComplexDouble | Сукупність тестів. Перевіряється правильність арифметичних дій над комплексними числами | Пройдено |
| DFMSignal | Сукупність тестів. Перевіряється правильність операцій додавання та множення сигналів | Пройдено |

3.2.2 Тести сценаріїв

Сценарій №1: застосування фільтру та сигналу за замовченням і одержання результату.

Вхідні дані:

Фільтр Identity має давати вихідний сигнал ідентичний вхідному. Коефіцієнти чисельника – {1}, знаменника – {1}.

Фільтр Average2 має усереднювати 2 попередні значення вхідного сигналу. Коефіцієнти чисельника – {0.5, 0.5}, знаменника – {1}.

Фільтр Delay має видавати вхідний сигнал з затримкою на 1. Коефіцієнти чисельника – {0, 1}, знаменника – {1}.

Сигнал Constant – сталий сигнал величини 4(4, 4, 4, …)

Сигнал Periodic2 – періодичний сигнал з періодом 2 і амплітудою 4(4, -4, 4, -4, …)

Сигнал Periodic4 – періодичний сигнал з періодом 4 і амплітудою 4(4, 0, -4,0, 4,0, -4, 0, 4, …)

Послідовність дій:

– запустити програму;

– обрати один з фільтрів за замовченням(Identity, Average2, Delay та інші);

– обрати один з сигналів за замовченням(Constant, Periodic2, Periodic4 та інші);

– при виборі сигналу чи фільтру графіки оновлюються з використанням нових даних.

Результат: сценарій виконано успішно.

Сценарій №2: завантаження фільтру з файлу.

Вхідні дані:

Фільтр має бути створено за правилам створення файлів, що описані вище.

Послідовність дій:

– запустити програму.

– натиснути кнопку Load filter from file

– обрати файл, в якому записано фільтр.

– при успішному зчитуванні файлу, фільтр автоматично застосовується до поточного сигналу і графіки оновлюються.

Результат: сценарій виконано успішно.

Сценарій №3: завантаження сигналу з файлу.

Послідовність дій:

– запустити програму.

– натиснути кнопку Load signal from file

– обрати файл, в якому записано сигнал.

– при успішному зчитуванні файлу, сигнал автоматично застосовується до поточного фільтру і графіки оновлюються.

Результат: сценарій виконано успішно.

Сценарій №4: збереження даних в файл.

Послідовність дій:

– запустити програму;

– застосувати довільний фільтр та довільний сигнал;

– натиснути кнопку Save signal to file;

– обрати, куди слід записати цей сигнал;

– вміст файлу має відповідати вихідному сигналу.

Результат: сценарій виконано успішно.

Сценарій №5: збереження сигналу в файл і подальше завантаження цього сигналу

Послідовність дій:

– запустити програму;

– застосувати довільний фільтр та довільний сигнал;

– натиснути кнопку Save signal to file;

– обрати, куди слід записати цей сигнал;

– кнопкою Load signal from file завантажити цей сигнал;

– сигнал має бути зчитано і застосовано до поточного фільтру.

Результат: сценарій виконано успішно.

### 3.2.3 Тести продуктивності

1. Розмір сигналу

За допомогою модифікаторів продовження створено сигнал довжини 4000.

Завантаження сигналу виконалося без помітних затримок.

Подальше тестування виконується окремо для степенів двійки і для інших чисел, оскільки для степенів двійки також виконується швидке перетворення Фурьє. Результати тестування для цих випадків наведені в таблицях 3.2 і 3.3.

Таблиця 3.2 – Тести довжини сигналу для довжин, що не є степенями двійки

| Довжина сигналу | Результат |
|---|---|
| 255 | Ледве помітна затримка при перерисуванні графіків(наприклад при зміні розмірів вікна). |
| 511 | Затримка є більш помітною, крім цього на графіку вже майже неможливо розрізнити власне сигнал. |
| 1023 | Затримка при зміні розмірів є значною, і також помітною стає затримка обчислення |
| 2047 | Обидві затримки є некомфортними для користування(більше секунди) |
| 4095 | Зміна розмірів графіків/вікна практично неможливими. Перерозрахунок займає декілька секунд. |

Таблиця 3.3 – Тести довжини сигналу для довжин, що є степенями двійки

| Довжина сигналу | Результат |
|---|---|
| 256 | Ледве помітна затримка при розрахунку, також спостерігається помітна затримка перерисування |
| 512 | Затримка розрахунку є значною, крім цього затримка зміни розмірів помітно більша ніж при довжині 511 |
| 1024 | Затримки приблизно такого ж порядку, як і для довжини 2047 |
| 2048 | Зміна розмірів є майже неможливою, аналогічно довжині 4095 |

Надалі ми будемо тестувати саме завантаження сигналів більшої довжини.

Завантаження сигналу довжини 10000 дає майже непомітну затримку.

Завантаження сигналу довжини 100000 дає затримку в декілька секунд.

Тестування більших значень не має сенсу, оскільки вже повна обробка сигналу довжини 100000 займає час порядку кілька десятків секунд навіть для найпростіших фільтрів.

Таким чином основним обмеженням для користування програмою є час обчислення та графічного відображення даних.

2. Порядок фільтру.

Оскільки порядок фільтру має сенс лише для сигналів з довжиною не меншою за порядок, створимо сигнал довжиною 1000 і будемо тестувати на ньому.

Порядок фільтру 10 не дає помітних затримок в порівнянні з фільтром порядку 1.

Порядок фільтру 20 дає ледве помітний приріст затримки.

## 3.3 Висновки до розділу

Було проведене тестування програми. Проводилося тестування 3 основними типами: юніт-тести, тести сценаріїв та тести продуктивності. Юніт-тести та тести сценаріїв дозволили знайти і виправити помилки в програмі. Результуюча програма проходить всі ці тести без помилок. Тести продуктивності дають наступні обмеження для розмірів сигналів: до 2000 без перетворення Фурьє і до 1000 з перетворенням для комфортної роботи з програмою, до 4000 без перетворення і до 2000 з перетворенням, якщо затримка є допустимою. Обмеження на порядок фільтру практично не мають значення, оскільки навіть для значних порядків(20) затримка є ледве помітною.

# 4 РОЗРОБЛЕННЯ ТЕХНІЧНОЇ ДОКУМЕНТАЦІЇ

## 4.1 Діаграма прецедентів

Діаграма прецедентів описує, які дії користувачі різних типів можуть виконувати з програмою. В даній програмі користувачі не розбиваються на категорії, всі користувачі можуть виконувати однакові дії.

Список можливих дій користувача:

– завантаження фільтру з файлу;

– завантаження сигналу з файлу;

– вибір одного з збережених фільтрів;

– вибір одного з збережених сигналів;

– перегляд результуючого сигналу;

– збереження результуючого сигналу в файл(з можливістю подальшого завантаження);

– перегляд частотних характеристик фільтру;

– збереження частотних характеристик фільтру в файл;

– перегляд частотних розподілів сигналів;

– збереження частотних розподілів сигналів в файл.

Крім цього є й інші дії, які може виконувати користувач(запустити/закрити програму, перемістити вікно, тощо), але вони не є особливими для даної програми і тому не будуть розглядатись.

В таблицях 4.1-4.10 наведений детальний опис варіантів використання. Він не є необхідним для побудови діаграми, але буде корисним для опису можливостей користувача в керівництві, а також при подальшому розвитку проекту.

Результуюча діаграма наведена в кресленику ІТ61.120БАК.004 Д2.

### 4.1.1 Завантаження фільтру з файлу

Таблиця 4.1 – Опис дії завантаження фільтру з файлу

| | |
|---|---|
| Мета | Підключити фільтр, який ще не було додано в програму |
| Вхідні дані | Файл, створений користувачем, що містить дані фільтру, та шлях до цього файлу |
| Вихідні дані | Результат застосування нового фільтру до поточного сигналу |
| Передумова | Файл створено користувачем, до файлу внесено необхідні дані |
| Успішний сценарій | Фільтр додається в програму і застосовується до поточного сигналу |
| Виключення | Файл не є правильно побудованим файлом з даними фільтру |
| Включає дії: | - |
| Включається діями: | - |
| Розширений діями: | - |
| Розширяє дії: | - |
| Додаткові відомості | Виконання цієї дії викликає перерозрахунок результуючого сигналу на заданій довжині, та перерисування графіків |

4.1.2 Завантаження сигналу з файлу

Таблиця 4.2 – Опис дії завантаження сигналу з файлу

| | |
|---|---|
| Мета | Підключити сигнал, який ще не було додано в програму |
| Вхідні дані | Файл, створений користувачем, що містить сигнал, та шлях до цього файлу |
| Вихідні дані | Результат застосування нового сигналу до поточного фільтру |
| Передумова | Файл створено користувачем, до файлу внесено необхідні дані |
| Успішний сценарій | Сигнал додається в програму і застосовується до поточного фільтру |
| Виключення | Файл не є правильно побудованим файлом з даними сигналу |
| Включає дії: | - |
| Включається діями: | - |
| Розширений діями: | - |
| Розширяє дії: | - |
| Додаткові відомості | Виконання цієї дії викликає перерозрахунок результуючого сигналу на заданій довжині, та перерисування графіків |

4.1.3 Завантаження збереженого фільтру

Таблиця 4.3 – Опис дії завантаження збереженого фільтру

| | |
|---|---|
| Мета | Підключити фільтр, що був попередньо збережений в програмі |
| Вхідні дані | Назва фільтру |
| Вихідні дані | Результат застосування нового фільтру до поточного сигналу |
| Передумова | Фільтр було попередньо завантажено з файлу, в файлі була вказана унікальна назва |
| Успішний сценарій | Фільтр застосовується до поточного сигналу |
| Виключення | Відсутні, неправильно побудований фільтр не може бути збережено в програмі |
| Включає дії: | - |
| Включається діями: | - |
| Розширений діями: | - |
| Розширяє дії: | - |
| Додаткові відомості | Виконання цієї дії викликає перерозрахунок результуючого сигналу на заданій довжині, та перерисування графіків |

4.1.4 Завантаження збереженого сигналу

Таблиця 4.4 – Опис дії завантаження збереженого сигналу

| | |
|---|---|
| Мета | Підключити сигнал, що був попередньо збережений в програмі |
| Вхідні дані | Назва сигналу |
| Вихідні дані | Результат застосування нового сигналу до поточного фільтру |
| Передумова | Сигнал було попередньо завантажено з файлу, в файлі була вказана унікальна назва |
| Успішний сценарій | Сигнал застосовується до поточного фільтру |
| Виключення | Відсутні, неправильно побудований сигнал не може бути збережено в програмі |
| Включає дії: | - |
| Включається діями: | - |
| Розширений діями: | - |
| Розширяє дії: | - |
| Додаткові відомості | Виконання цієї дії викликає перерозрахунок результуючого сигналу на заданій довжині, та перерисування графіків |

4.1.5 Перегляд результуючого сигналу

Таблиця 4.5 – Опис дії перегляду результуючого сигналу

| | |
|---|---|
| Мета | Графічно переглянути результат дії фільтру на сигнал |
| Вхідні дані | - |
| Вихідні дані | Графік, на якому зображено результуючий сигнал в порівнянні з вхідним |
| Передумова | Було встановлено необхідний вхідний сигнал і фільтр, а також обрана довжина |
| Успішний сценарій | Виведено відповідний графік |
| Виключення | - |
| Включає дії: | - |
| Включається діями: | - |
| Розширений діями: | Збереження результуючого сигналу |
| Розширяє дії: | - |
| Додаткові відомості | Якщо в програмі увімкнуті інші графіки, графік сигналів слід увімкнути, обравши будь-який графік і перемкнувши його в режим графіку сигналів |

4.1.6 Збереження результуючого сигналу

Таблиця 4.6 – Опис дії збереження результуючого сигналу

| | |
|---|---|
| Мета | Зберегти результуючий сигнал в файл |
| Вхідні дані | - |
| Вихідні дані | Файл, створений в обраному користувачем місці, в якому записано результуючий сигнал |
| Передумова | Було встановлено необхідний вхідний сигнал і фільтр, а також обрана довжина |
| Успішний сценарій | Створено відповідний файл |
| Виключення | Користувач відмінив операцію створення файлу, чи вказав некоректний шлях чи назву.<br>Користувач має недостатньо прав для створення файлів в обраному місці.<br>На обраному диску недостатньо місця. |
| Включає дії: | - |
| Включається діями: | - |
| Розширений діями: | Завантаження отриманого сигналу в програму(не розглянуто окремо, див. таблицю 4.2) |
| Розширяє дії: | Перегляд результуючого сигналу |
| Додаткові відомості | Сигнал створено з назвою за замовченням. За необхідності, її можна змінити, відредагувавши файл. |

4.1.7 Перегляд частотних характеристик фільтру

Таблиця 4.7 – Опис дії перегляду характеристик фільтру

| | |
|---|---|
| Мета | Графічно переглянути характеристики фільтру |
| Вхідні дані | - |
| Вихідні дані | Графік, на якому зображено обрану характеристику як графік залежності від частоти |
| Передумова | Було встановлено необхідний фільтр, а також обрана довжина |
| Успішний сценарій | Виведено відповідний графік |
| Виключення | - |
| Включає дії: | - |
| Включається діями: | - |
| Розширений діями: | Збереження характеристик фільтру |
| Розширяє дії: | - |
| Додаткові відомості | Якщо в програмі увімкнуті інші графіки, графік обраної характеристики слід увімкнути, обравши будь-який графік і перемкнувши його в режим графіку обраної характеристики фільтру |

4.1.8 Збереження частотних характеристик фільтру

Таблиця 4.8 – Опис дії збереження характеристик фільтру

| | |
|---|---|
| Мета | Зберегти обрану характеристику фільтру в файл |
| Вхідні дані | - |
| Вихідні дані | Файл, створений в обраному користувачем місці, в якому записано частотну характеристику |
| Передумова | Було встановлено необхідний фільтр, а також обрана довжина |
| Успішний сценарій | Створено відповідний файл |
| Виключення | Користувач відмінив операцію створення файлу, чи вказав некоректний шлях чи назву.<br>Користувач має недостатньо прав для створення файлів в обраному місці.<br>На обраному диску недостатньо місця. |
| Включає дії: | - |
| Включається діями: | - |
| Розширений діями: | - |
| Розширяє дії: | Перегляд характеристик фільтру |
| Додаткові відомості | Файл неможливо відкрити програмою, функція запису створена для більш зручного аналізу отриманої характеристики в інших програмах |

4.1.9 Перегляд частотних розподілів сигналів

Таблиця 4.9 – Опис дії перегляду частотних характеристик сигналів

| | |
|---|---|
| Мета | Графічно переглянути характеристики сигналів |
| Вхідні дані | - |
| Вихідні дані | Графік, на якому зображено обрану характеристику обох сигналів для порівняння |
| Передумова | Було встановлено необхідний вхідний сигнал і фільтр, а також обрана довжина |
| Успішний сценарій | Виведено відповідний графік |
| Виключення | - |
| Включає дії: | - |
| Включається діями: | - |
| Розширений діями: | Збереження характеристик сигналів |
| Розширяє дії: | - |
| Додаткові відомості | Якщо в програмі увімкнуті інші графіки, графік обраної характеристики слід увімкнути, обравши будь-який графік і перемкнувши його в режим графіку обраної характеристики сигналів |

4.1.10 Збереження частотних характеристик фільтру

Таблиця 4.10 – Опис дії збереження характеристик фільтру

| | |
|---|---|
| Мета | Зберегти обрану характеристику сигналу в файл |
| Вхідні дані | - |
| Вихідні дані | Файл, створений в обраному користувачем місці, в якому записано частотну характеристику сигналу |
| Передумова | Було встановлено необхідний вхідний сигнал і фільтр, а також обрана довжина |
| Успішний сценарій | Створено відповідний файл |
| Виключення | Користувач відмінив операцію створення файлу, чи вказав некоректний шлях чи назву.<br>Користувач має недостатньо прав для створення файлів в обраному місці.<br>На обраному диску недостатньо місця. |
| Включає дії: | - |
| Включається діями: | - |
| Розширений діями: | - |
| Розширяє дії: | Перегляд частотних характеристик сигналів |
| Додаткові відомості | Файл неможливо відкрити програмою, функція запису створена для більш зручного аналізу отриманої характеристики в інших програмах<br>В файл можна записати лише одну характеристику одного сигналу |

## 4.2 Діаграма активності

Діаграма активності зображує процеси, що відбуваються в програмі при виконанні користувачем певного сценарію. Побудова цієї діаграми є корисною для розробки та тестування програмного забезпечення, а також для виявлення помилок.

В даній роботі ми побудуємо діаграми двох основних активностей: підключення фільтру/сигналу, та збереження даних в файл.

Хоча фільтри і сигнали є принципово різними складовими програми, ця різниця представлена здебільшого в методах обчислення та в методах завантаження/запису, і більшість операцій, вказаних на діаграмі, виконуються для них однаково. Надалі ми будемо використовувати «фільтр», враховуючи, що сигнал підключається аналогічно, а, у випадку наявності різниці, ця різниця буде вказана окремо.

Результуюча діаграма наведена на кресленику ІТ61.120БАК.004 Д4.

### 4.2.1 Підключення фільтру чи сигналу

Якщо у користувача виникає необхідність підключити фільтр, перше питання, яке слід задати – чи був цей фільтр вже завантажено в програму. Переглянути список завантажених фільтрів користувач може за допомогою випадаючого списку. Якщо фільтр присутній в списку, його можна відразу обрати там. Якщо фільтр відсутній, його слід завантажити з файлу, вказавши файл в діалоговому вікні.

Незалежно від цього вибору, обраний фільтр слід прочитати з файлу. Команда користувача отримується View, і View передає відповідному контролеру команду завантаження і шлях до файлу.

Після того, як контролер прочитає файл, одержаний фільтр(у випадку успішного зчитування) застосовується в ViewModel. Це призводить до

виклику команди в Model, що виконує переобчислення даних. Model повертає пере обчислені дані у вигляді масивів комплексних чисел.

Оскільки одна з задач ViewModel – привести дані в зручний для View вигляд, виконується розбиття масивів комплексних чисел на масиви дійсних. Всі ці масиви зберігаються під окремими назвами. Після цього, ViewModel викликає подію оновлення, на яку підписаний View.

View, при виклику цієї події, зчитує необхідні дані з ViewModel і будує за ними графіки. Слід вказати, що аналогічне перерисування викликається також при перемиканні режимів графіків чи при зміні їх розмірів, але в цих випадках перерисування керується безпосередньо в View.

Користувач отримує оновлені графіки на екрані, дію завершено.

### 4.2.2 Збереження даних в файл

Програма дозволяє зберігати наступні дані:

- Вихідний сигнал;
- АЧХ фільтру;
- ФЧХ фільтру;
- Амплітудний розподіл вхідного сигналу за частотою;
- Фазовий розподіл вхідного сигналу за частотою;
- Амплітудний розподіл вихідного сигналу за частотою;
- Фазовий розподіл вхідного сигналу за частотою.

Хоча на перший погляд всі ці дані виглядають абсолютно різними, з точки зору запису в файл вони всі представляють собою послідовність чисел. Крім цього, ці дані вже присутні на момент початку запису, оскільки вони обчислюються в Model відразу після застосування нового фільтру чи сигналу, і можуть виводитись графічно. Через це ми надалі будемо говорити про «дані», враховуючи що вони можуть бути одного з наведених типів.

Користувач починає активність з натискання кнопки збереження даних. В залежності від того, що саме було обрано для збереження, View викликає відповідну команду контролера файлів, передавши туди масив значень, зчитаний з ViewModel. Крім цього, при натисканні кнопки, View пропонує користувачу обрати, куди саме слід записати дані, і передає контролеру шлях.

Контролер не обов’язково має бути саме контролером файлів, але в даній роботі інші контролери не реалізовано. Результат запису (успіх/неуспіх+причина) передаються в ViewModel, який, в свою чергу, викликає подію, на яку підписано View.

View, при виклику цієї події, виводить користувачу відповідне повідомлення, дію завершено.

# 5 КЕРІВНИЦТВО КОРИСТУВАЧА

## 5.1 Впровадження і супровід програмного забезпечення

Робота ПЗ грунтується на встановленій .Net Framework 4.6, тому її слід встановити. Після цього програма запускається при відкриванні файлу DFM.exe. Для коректної роботи програми слід помістити її в окрему директорію. Разом з програмою постачаються директорії з прикладами сигналів та фільтрів, які слід помістити в директорію з програмою.

ПЗ не потребує додаткового супроводу.

## 5.2 Опис інтерфейсу

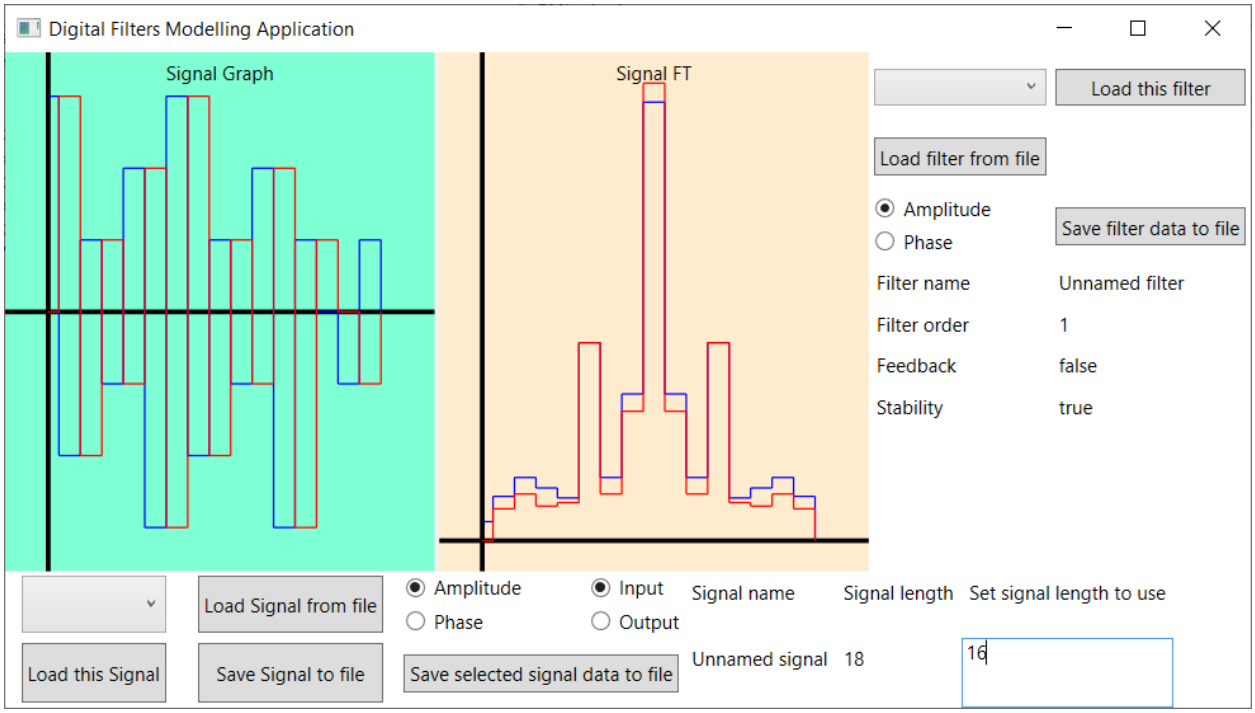


Рисунок 5.1 – Інтерфейс програми DFM Application

На рисунку 5.1 зображено інтерфейс програми. Він складається з 3 основних частин: графічна область, область роботи з фільтрами і область роботи з сигналами.

Графічна область складається з 2 графіків. Переміщуючи лінію розділу між ними можна виділити більше місця до одного з них за рахунок іншого.

За замовченням на першому графіку зображуються вхідний сигнал(синій графік) і вихідний сигнал(червоний графік). На другому сигналі зображуються амплітудно-частотні характеристики сигналів(для довжин що є степенями двійки). Будь-який графік можна перемкнути на один з режимів:

– зображення сигналів;

– зображення амплітудно-частотних характеристик сигналів;

– зображення фазово-частотних характеристик сигналів;

– зображення амплітудно-частотної характеристики фільтру;

– зображення фазово-частотної характеристики фільтру.

Область роботи з фільтрами включає в себе елементи керування фільтрами:

- кнопки для вибору фільтру за замовченням;
- кнопку для завантаження фільтру з файлу;
- кнопки для збереження частотних характеристик фільтру в файл,

а також інформаційні елементи, що відображують:

- назву фільтру;
- порядок фільтру;
- рекурсивність і стійкість.

Область роботи з сигналами включає в себе елементи керування сигналами:

- кнопки для вибору сигналу за замовченням;
- кнопку для завантаження сигналу з файлу;
- кнопку для збереження результуючого сигналу в файл;
- кнопки для збереження частотних характеристик сигналів в файл;
- поле вибору робочої довжини сигналу

а також інформаційні елементи, що відображують

- назву сигналу;

– довжину сигналу.

## 5.3 Інструкції по роботі з програмою

### 5.3.1 Робота з сигналами

Можна використовувати вбудовані сигнали або додавати сигнали з файлів. Щоб застосувати вбудований сигнал, слід обрати його в списку вбудованих сигналів, і потім натиснути кнопку «Load this signal». Щоб завантажити сигнал з файлу, слід натиснути кнопку «Load signal from file», знайти відповідний файл і обрати його. При успішному завантаженні файлу копія сигналу буде додана в директорію програми «Saved signals», і, після цього, цей сигнал буде присутній в списку вбудованих сигналів.

Обраний сигнал одразу подається на активний фільтр, і результати виводяться на графіках.

За допомогою поля Set signal length to use можна виставити довжину сигналів, яка буде використовуватись фільтром. Якщо довжина сигналу менша за це значення, решта значень буде вважатись нульовими.

Вихідний сигнал можна зберегти в файл кнопкою «Save signal to file», після чого програма запропонує користувачу обрати, куди слід записати файл. Отриманий файл бажано відредагувати, вписавши туди назву сигналу.

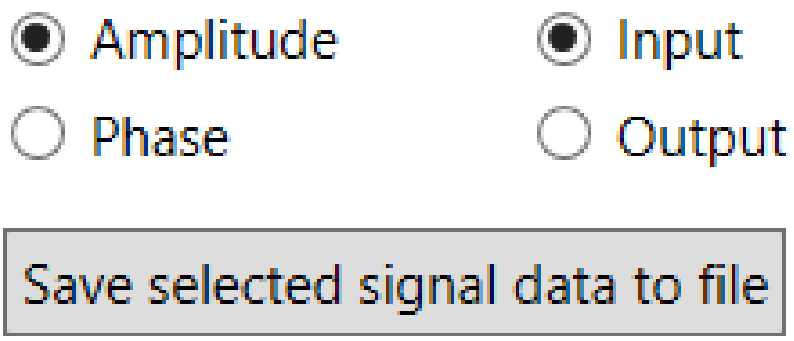


Рисунок 5.2 – Перемикачі і кнопка, що відповідають за збереження даних сигналів в файл

Крім самого сигналу, в файл можна зберегти частотні характеристики, як вхідного, так і вихідного сигналу. Для цього слід обрати довжину – степінь двійки, обрати, яку саме характеристику(амплітудну чи фазову) і якого сигналу(вхідного чи вихідного) слід записати(використавши перемикачі, зображені на рис. 5.2), натиснути кнопку «Save signal values», після чого програма запропонує користувачу обрати, куди слід записати дані. На відміну від записаних сигналів, записані характеристики неможливо прочитати програмою.

Натиснувши на область одного з графіків, можна переключити його для зображення самих сигналів, чи одної з їх характеристик.

### 5.3.2 Робота з фільтрами

Як і сигнали, фільтри можна використовувати як вбудовані, так і завантажені з файлів. Щоб застосувати вбудований фільтр, слід обрати його в списку вбудованих фільтрів, і потім натиснути кнопку «Load this filter». Щоб завантажити фільтр з файлу, слід натиснути кнопку «Load filter from file», знайти відповідний файл і обрати його. При успішному завантаженні файлу копія фільтру буде додана в директорію програми «Saved filters», і, після цього, цей фільтр буде присутній в списку вбудованих фільтрів.

Обраний фільтр одразу застосовується до поточного сигналу, результати виводяться на графіках.

Можна записати амплітудну чи фазову характеристику фільтра в файл, обравши відповідну характеристику та натиснувши кнопку «Save filter values», після чого користувачу буде запропоновано обрати, куди слід записати дані. Як і з даними сигналів, дані фільтрів неможливо прочитати програмою.

Крім запису в файл, характеристики фільтра можна переглянути в програмі, натиснувши на один з графіків і перемкнувши його в режим перегляду відповідної характеристики.

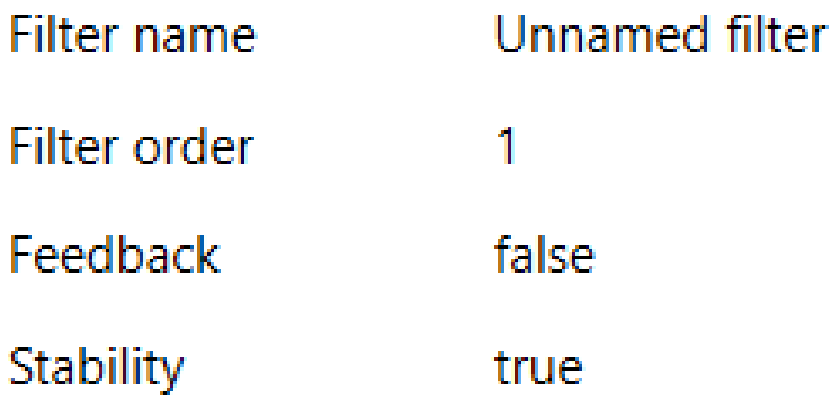


Рисунок 5.3 Додаткові відомості про фільтр

Крім цього, як показано на рис. 5.3, в програмі можна переглянути додаткові відомості про обраний фільтр:

– Filter name – назва фільтру(взята з файлу);

– Filter order – порядок фільтру(максимальна степінь при $z^{-1}$ в передавальній функції, або, що те ж саме, скільки попередніх значень вхідного чи вихідного сигналу необхідно для обчислення поточного значення);

– Feedback – чи присутній в фільтрі зворотній зв'язок. Фільтри без зворотнього зв’язку залежать лише від значень вхідного сигналу, а фільтри зі зворотнім зв’язком – ще й від значень вихідного сигналу;

– Stability – чи наближається імпульсний відгук даного фільтру до 0. Будь-який фільтр без зворотнього зв’язку є стабільними. Нестабільні фільтри можуть давати незатухаючі сигнали навіть якщо вхідний сигнал закінчився необмежено давно.

Частотні характеристики фільтра можна порівняти з «експериментальними», обчисленими з порівняння вхідного та вихідного сигналів, відкривши їх на графіках.

## 5.4 Висновки до розділу

Хоча одною з основних вимог до програми є побудова інтуїтивно зрозумілого інтерфейсу, керівництво користувача є однією з необхідних складових для роботи з програмним забезпеченням. В даному керівництві було описано інтерфейс програми та його основні частини, та розглянуто основні дії, які може виконати користувач.

# ВИСНОВКИ

Результатом розробки є програмне забезпечення для моделювання цифрових фільтрів. Ця програма дозволяє завантажувати фільтри та сигнали, переглядати результат дії фільтру на сигнал та зберігати результат. Формат вхідних файлів дозволяє зручно створювати сигнали та фільтри без використання додаткового програмного забезпечення. Крім цього, є можливість перегляду та збереження частотних характеристик фільтру та сигналів.

В ході виконання дипломного проєкту було побудовано десктоп додаток, що має зручний інтерфейс і наочний метод зображення даних. Додаток побудовано згідно з вимогами, поставленими після аналізу предметної області і порівняння програм аналогів. Крім цього, додаток побудовано з врахуванням сучасних вимог до архітектури програмного забезпечення.

Планується подальша робота над проєктом та його вдосконаленням. Для розвитку даної проекту необхідно додати ряд нових функцій, присутніх в програмах-аналогах, а саме генерація фільтрів стандартних типів та можливість редагування фільтрів та сигналів в програмі. Також можливе покращення графічного інтерфейсу і додавання користувацьких налаштувань, а також розширення можливостей зовнішнього збереження даних.

В роботі описано процес аналізу предметної області, побудова вимог до програми, проектування та розробки програмного забезпечення, вибір основних методів та інструментів для реалізації даного продукту. Було обрано метод збереження даних в файлах. Була детально описана побудова діаграми прецедентів і діаграми активності.

Дана система була протестована на правильність виконання окремих функцій, правильність виконання сценаріїв роботи та швидкодію при навантаженнях. Планується подальше тестування програми.

Для впровадження та супроводу програмного забезпечення було розроблено детальну інструкцію з підготовки до встановлення, встановлення та користування програмним забезпеченням.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Цифровой сигнал. URL: https://ru.wikipedia.org/wiki/Цифровой_сигнал

2. Дискретизация. URL: https://ru.wikipedia.org/wiki/Дискретизация

3. Квантование_(обработка_сигналов). URL: https://ru.wikipedia.org/wiki/Квантование_(обработка_сигналов)

4. Комплексное число. URL: https://ru.wikipedia.org/wiki/Комплексное_число

5. Digital_filter. URL: https://en.wikipedia.org/wiki/Digital_filter

6. Z-Transform. URL: https://mathworld.wolfram.com/Z-Transform.html

7. Тема 9. ЦИФРОВЫЕ ФИЛЬТРЫ URL: https://portal.tpu.ru/SHARED/v/VOS/study/disc1/Tab/tema09.pdf

8. Гармонический сигнал URL: https://ru.wikipedia.org/wiki/Гармонический_сигнал

9. Discrete Fourier transform URL: https://en.wikipedia.org/wiki/Discrete_Fourier_transform

10. Development of a Fourier transform in C# URL: https://www.egr.msu.edu/classes/ece480/capstone/fall11/group06/style/Application_Note_ChrisOakley.pdf

11. 1-D digital filter – MATLAB filter URL: https://www.mathworks.com/help/matlab/ref/filter.html

12. Pricing and Licensing - MATLAB & Simulink URL: https://www.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=comm

13. Digital Filter Design Software Fiwiz URL: https://www1.icsi.berkeley.edu/~storn/fiwiz.html

14. FilterLab Filter Design Software URL: https://www.microchip.com/developmenttools/ProductDetails/filterlabdesignsoftwfil#additional-summary

15. Model-View-ViewModel URL: https://ru.wikipedia.org/wiki/Model-View-ViewModel

16. .NET Framework URL: https://en.wikipedia.org/wiki/.NET_Framework

17. Windows Presentation Foundation URL: https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation